Implementación de juegos de estrategia en tablero
SPECULATE - Memoria del proyecto

Javier Segura

13 de enero de 2015

Datos del proyecto

Título del proyecto: Implementación de juegos de estrategia en tablero - SPECULATE
Nombre del estudiante: Javier Segura Martínez
Titulación: Ingeniería Informática (2003)
Créditos: 37.5
Director/Ponente: Miguel Barceló Garcia
Departamento: Ingeniería de Servicios y Sistemas de Información (ESSI)

Miembros del tribunal

Presidente: Xavier Burgués Illa

Vocal: Klaus Gerhard Langohr

Secretario: Miquel Barceló Garcia

Calificación

Calificación numérica:
Calificación descriptiva:
Fecha: 21 de Enero de 2015

Agradecimientos

Me gustaría dedicar esta memoria a mi padres.
A mi padre, Juan, por su tenacidad, perseverancia y paciencia.
A mi madre, Ana, por su cariño y comprensión.
Sin ellos no podría haber llegado a ser lo que soy hoy en día.
Dedicado a vosotros.

Índice general

1. Introducción **11**
1.1. Speculate: el juego original 11
1.2. Speculate: implementación online 14

2. Motivación y objetivos **15**
2.1. Contexto 15
2.2. Motivación 15
2.3. Objetivos 16

3. Análisis y especificación **17**
3.1. Requisitos 17
3.1.1. Requisitos funcionales 17
3.1.2. Requisitos no funcionales 18
3.2. Modelo de casos de uso 18
3.2.1. Listado de casos de uso 19
3.3. Mapa conceptual de los datos 22

4. Diseño e implementación **25**
4.1. Wireframing y prototipo 25
4.1.1. Resultado final 29
4.2. Entorno y plataforma 37
4.2.1. Cliente web 37
4.2.2. Servidor 38
4.2.3. Capa de datos 38
4.2.4. Librerías utilizadas 39
4.2.5. Frontend 39
4.2.6. Backend 39
4.3. Estados de un juego 40
4.4. Implementación de la IA 40
4.5. Implementación REST de backend 41
4.6. Autenticación de los clientes 42

4.7. Implementación de la comunicación en tiempo real 42
4.7.1. Uso dentro de Speculate 42

5. Pruebas **45**
5.1. Tests unitarios 45
5.2. Tests manuales 46

6. Planificación y estudio económico **49**
6.1. Planificación 49
6.1.1. Estimación 49
6.1.2. Realidad 50
6.1.3. Desviación de la planificación inicial 50
6.2. Análisis económico 50
6.2.1. Horas invertidas 51
6.2.2. Coste de equipamiento 51

7. Conclusiones **53**

Apéndices **55**

A. Glosario **57**

B. Bibliografía **59**

C. Manual del juego original **61**
C.1. Manual original 61
C.1.1. The aim of the game 61
C.1.2. Strategy 61
C.1.3. Equipment 62
C.1.4. Preparation 62
C.1.5. Duration of the game 63
C.1.6. Winning the game 63
C.1.7. Objectives 63
C.1.8. The turn 64
C.1.9. Buying 64
C.1.10. Selling 65
C.1.11. First and last rounds 65
C.1.12. End of the round 65
C.1.13. Share prices 66
C.1.14. Dividend payments 66
C.1.15. Bankruptcy 66
C.2. Manual traducido al castellano 66
C.2.1. El objetivo del juego 66
C.2.2. Estrategia 67
C.2.3. Equipo 67
C.2.4. Preparación 67
C.2.5. Duración del juego 68
C.2.6. Ganar el juego 68
C.2.7. Objetivos 68
C.2.8. El Turno 69
C.2.9. Compra 70

C.2.10. Venta 70
C.2.11. Rondas primera y la última 71
C.2.12. Fin de la ronda 71
C.2.13. Precios de las acciones 71
C.2.14. El pago de dividendos 71
C.2.15. Bancarrota 72

D. Código abierto **73**
D.1. Licencia 73
D.2. Código fuente 73
D.2.1. Cómo obtener y ejecutar el código 74

1 Introducción

Este proyecto de final de carrera presenta el análisis, el diseño y la implementación de Speculate, un juego de mesa por turnos creado en 1972 por *Graeme Levin*, usando tecnologías actuales. A continuación describiré en detalle el juego original y su implementación online.

1.1. Speculate: el juego original

Speculate es un juego sobre el mercado de valores. El objetivo principal es conseguir ser el jugador más rico al final de todos los turnos de la partida. Para ésto se suman tanto el valor de las acciones en posesión como la cantidad de dinero en efectivo que posean los jugadores.

Figura 1.1: Juego original

Hay determinados factores que caracterizan al juego. Aquí presento una lista de elementos del juego con sus respectivas imágenes [a]:

- Al principio del juego cada jugador obtiene un *objetivo oculto*; dicho objetivo consiste en poseer cierta cantidad de dinero en unas determinadas acciones y no poseer ningún título de otros tipos de acciones. Ésto hace que haya un componente especial de diversión, ya que esos objetivos son ocultos y los jugadores deciden si quieren conseguirlos o no. A su vez, el hecho de que haya jugadores que quieran cumplir sus objetivos hace que haya cierta actividad en el juego.

[a] Imágenes extraidas de [1, BoardGeekGame]

Figura 1.2: Cartas de objetivo

- A parte de los objetivos, los jugadores reciben una serie de cartas de influencia con las cuales pueden cambiar el comportamiento del mercado. Las cartas de influencia pueden subir o bajar el precio de las acciones o repartir dividendos entre todos los accionistas. Al jugar una carta de objetivo, se podrá, indirectamente, beneficiar o perjudicar al resto de jugadores. Dichas cartas, junto con la carta de objetivo, son ocultas.

Credit restrictions eased | Restrictions de crédit atténuées
Stores | Grands Magasins
up | hausse
4
Credit restrictions imposed | Restrictions de crédit imposées
All Shares | Toutes Actions
down | baisse
1
Property | Propriétés
Dividend | Dividende
A
Shares at
B
Per share
0-4 pay 0
5-12 " 1
13-20 " 3
21-30 " 5
Find market price in Col. A. Take related figure in Col. B. and multiply by number of shares held for total dividend.
A
Actions à
B
Par action
0-4 payer 0
5-12 " 1
13-20 " 3
21-30 " 5
Pour le dividende total, trouver le cours du marché dans les colonnes A prendre le chiffre correspondant dans la colonne B et multiplier par le nombre d'actions en possessions.

Figura 1.3: Cartas de influencia

- También se pueden realizar compra y venta de títulos. Las posibles acciones a realizar se pueden efectuar contra el resto de jugadores o contra el Banco. Cabe decir, que las acciones en las que el Banco es la contraparte, tienen cierta penalización sobre el precio de la operación. Es decir, si quiero vender unos títulos de acciones que poseo y el banco es quién me los compra, lo hará a un precio más bajo que el precio de mercado.

Figura 1.4: Títulos de acciones

1.2. Speculate: implementación online

Cómo se ha descrito anteriormente en este capítulo, este proyecto presenta una implementación moderna del juego usando tecnologías actuales. Se ha implementado una versión del juego capaz de ser jugada online, habilitando así el poder usarse en cualquier dispositivo actual como pueden ser ordenadores, tablets o smartphones. Se ha hecho especial énfasis sobre la escalabilidad del mismo y en la optimización para poder utilizarse sobre redes móviles. Se hace un análisis más detallado sobre el diseño del juego en el Capítulo 4.

2 Motivación y objetivos

2.1. Contexto

En el 2013 se han reportado unos 2,8 billones de usuarios de internet en todo el mundo según [2, Internet World Stats]. Éste ha sido uno de los motivos fundamentales para seleccionar la tecnología para implementar el juego, básicamente porque la escalabilidad es un factor muy a tener en cuenta. Si cualquier aplicación de hoy en día tiene éxito, seguramente se enfrentará a problemas de rendimiento debido a la cantidad de posibles usuarios.

Desarrollar aplicaciones escalables se ha convertido en uno de los retos comunes hoy en día en cualquier desarrollo. En la implementación de Speculate se han seleccionado tecnologías que se puedan escalar horizontalmente, ya sea en diferentes servidores o incluso en diferentes Datacenters.

Sin olvidar comentar que la industria de los videojuegos es una gran industria que seguirá creciendo hasta alcanzar los 100.000 millones de dólares en 2017 según [3, DigitalCapital].

2.2. Motivación

Los datos descritos en el apartado anterior ya son de por si una gran motivación para realizar este proyecto. Pero uno de los principales factores que me ha hecho decidirme para implementar el juego de manera online, ha sido, por una parte, la temática del juego, y por otra la posibilidad de realizar una implementación moderna usando tecnologías actuales.

- *Temática*: El mundo bursátil siempre ha sido un tema que me ha apasionado. El poder programar un juego basado en esa temática es un evento que no era viable desperdiciar.

- *Nuevas tecnologías*: La idea de poder recrear un juego de hace más de 30 años en un entorno moderno siempre es un reto apasionante. Se pueden adaptar nuevas tecnologías, en este caso un entorno web y comunicación con websockets, para poder intentar plasmar un juego de mesa en un entorno totalmente virtual.

Por otro lado, gracias a esta implementación se puede disfrutar del juego sin necesidad de esperar a que otros jugadores quieran participar. Se han implementado diversos algoritmos para dotar de inteligencia artificial a jugadores ficticios.

La implementación de este juego me ha servido como excusa para iniciarme en un campo de la informática sobre el que no tengo ninguna experiencia profesional. Entornos web con Javascript tanto en el cliente como en el servidor, bases de datos NOSQL, comunicación en tiempo real mediante Websockets.

Todo ello usando la filosofía de código abierto, devolviendo así a la comunidad el conocimiento adquirido mediante el desarrollo del juego.

2.3. Objetivos

Una vez claros los motivos que me han llevado a la ejecución de este proyecto, ya podemos identificar los objetivos que se quieren conseguir con la ejecución del mismo.

1. Diseñar e implementar el juego desde el diseño inicial hasta la puesta en marcha en producción.
2. Plasmar la esencia del juego de mesa en la versión online implementada
3. Poder jugar contra el ordenador en la versión online implementada
4. Publicar el juego mediante una licencia de código abierto

Por otro lado, aquí hay una lista de objetivos secundarios, aunque no menos importantes, del proyecto:

1. Adquirir conocimiento de las nuevas vertientes en desarrollo web
2. Poder ejecutar el juego en otros dispositivos diferentes al ordenador personal
3. Diseñar una arquitectura escalable que pueda aguantar un crecimiento masivo de usuarios
4. Implementar el juego en varios idiomas

3

Análisis y especificación

La especificación de un proyecto consiste en definir con claridad cuales son las partes esenciales que se van a desarrollar y, sobre todo, qué funcionalidad van a ofrecer.
Para cumplir este propósito he divido la especificación en 3 bloques fundamentales:

1. Análisis de requisitos
2. Casos de uso
3. Modelo conceptual de los datos

3.1. Requisitos

Los requisitos en el ámbito del desarrollo de software se pueden dividir en dos grandes bloques: funcionales y no funcionales. A continuación vamos a listar todos los requisitos con una breve explicación de la importancia de los mismos.

3.1.1. Requisitos funcionales

Los requisitos funcionales son aquellos que tratan sobre el comportamiento del sistema, evaluando qué comportamiento se espera (salida) a partir de una interacción del actor (entrada). A partir de estos requisitos se definirán los casos de uso del sistema.

1. Los jugadores han de poder registrarse en el juego.
2. Los jugadores han de poder loguearse en el sistema.
3. Los jugadores han de poder crear partidas seleccionando entre diferentes tipos de Inteligencia Artifical
4. Los jugadores han de poder borrar partidas
5. Dada una partida, el jugador puede ejecutar las acciones básicas que el juego real le permite

a) Pagar al banco

b) Jugar una carta de influencia

c) Jugar una carta de objetivo

d) Empezar una compra/venta de acciones

e) Responder a una compra/venta de acciones de otros jugadores

3.1.2. Requisitos no funcionales

Los requisitos no funcionales son aquellos que no afectan directamente al comportamiento del sistema, pero pueden provocar modificaciones o restricciones en el diseño o la implementación.

1. **Usabilidad**: La implementación ha de ser usable por personas no técnicas. Cualquier persona con un conocimiento mínimo sobre la web debería ser capaz de poder jugar a la versión implementada.

2. **Escalabilidad**: La escalabilidad de un sistema se define a través de la facilidad del mismo para soportar grandes cantidades de carga de trabajo. Como ya se ha comentado anteriormente, el hecho de que el público objetivo sea una masa de jugadores tan grande, hace que a la hora de diseñar el sistema se tenga muy en cuenta la escalabilidad.

3. **Extensibilidad**: La extensibilidad de un sistema se define como la facilidad para agregar nuevos componentes al mismo. El proyecto se ha de desarrollar siguiendo estandards y metodologías comunmente aceptadas.

4. **Distribución y portabilidad**: Este requisito es una petición del director del proyecto, Miquel Barceló. El juego ha de poderse jugar sin que para ello necesitemos ejecutar complejos pasos para poder instalarlo.

5. **Multiidioma**: El juego ha de estar traducido a como mínimo 3 idiomas: Castellano, catalán y Inglés.

3.2. Modelo de casos de uso

Una vez listados los requisitos funcionales del sistema, se puede elaborar una lista de casos de uso que se ha plasmado en el siguiente gráfico. Como comentario, he de decir que no es un diagrama de casos de uso típico donde se pueden ver todas las acciones a realizar. En este caso he querido plasmar gráficamente los casos de uso y un tipo de modelo de estados en el mismo gráfico, donde se pueden ver qué otros casos de uso son aplicables al estado que obtenemos de cada ejecución. La decisión de usar este tipo de gráfico ha sido una decisión práctica, ya que considero que realizar muchos gráficos cuando toda la información puede plasmarse en un solo gráfico es contraproducente. De esta manera podemos hacernos una idea del comportamiento de la aplicación en un solo vistazo.

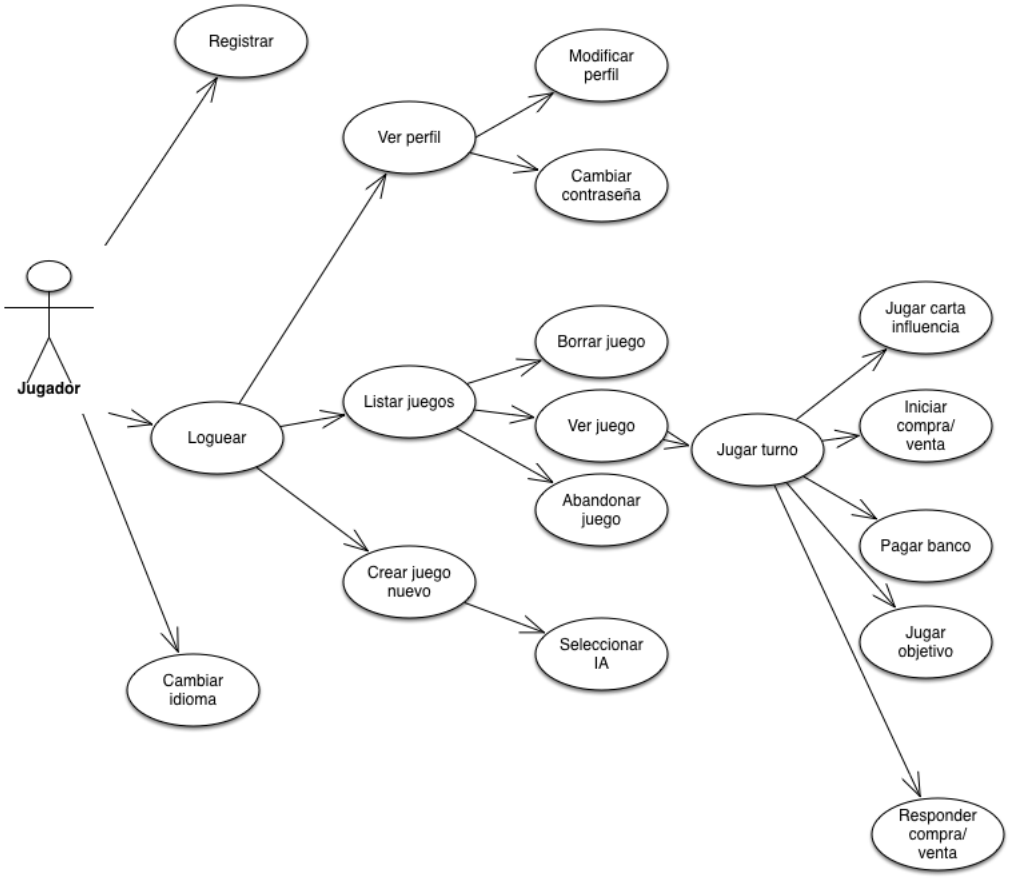

Figura 3.1: Casos de uso y estado de la aplicación

3.2.1. Listado de casos de uso

A continucación se listan los casos de uso identificados en el análisis explicados, incluyendo cuales son los datos de entrada y el comportamiento que ha de tener el sistema.

Nota: Para simplificar no se ha incluido el actor del sistema ya que es común a todos los casos de uso: el usuario de la aplicación.

Registrar

- **Descripción**: Cualquier usuario puede registrarse como jugador en el sistema. Ésto le dará derecho a poder usar el juego. Los datos introducidos serán validados
- **Parámetros de entrada**: email, nombre, contraseña.
- **Salida**: El usuario obtiene las credenciales para poder loguearse en el sistema.

Cambiar idioma

- **Descripción**: Cualquier usuario puede seleccionar el idioma mediante un selector.
- **Parámetros de entrada**: Idioma que se desea mostrar: Castellano, catalán o inglés.
- **Salida**: La interfaz visual del juego se cambia sin refrescar la página mostrando todos los textos en el idioma seleccionado.

Loguear

- **Descripción**: Cualquier usuario puede identificarse como jugador dentro del sistema
- **Parámetros de entrada**: email, contraseña
- **Salida**: Si las credenciales son correctas, el jugador obtendrá su lista de juegos, en caso contrario se le volverán a pedir las credenciales.

Ver perfil

- **Descripción**: Cualquier usuario logueado en el sistema puede acceder a su perfil
- **Parámetros de entrada**: Usuario logueado.
- **Salida**: El usuario podrá ver su imágen de Gravatar [4], nombre, email y cual fue su fecha de registro.

Modificar perfil

- **Descripción**: Cualquier usuario logueado puede cambiar sus datos de perfil
- **Parámetros de entrada**: nuevo nombre, email
- **Salida**: El jugador ha cambiado el nombre o email registrado en el sistema

Cambiar contraseña

- **Descripción**: Cualquier usuario logueado puede cambiar su contraseña
- **Parámetros de entrada**: contraseña antigua y actual
- **Salida**: El usuario puede loguearse usuando la nueva contraseña

Listar juegos

- **Descripción**: Cualquier usuario logueado puede ver la lista de juegos del sistema
- **Parámetros de entrada**: -
- **Salida**: Lista de juegos que están siendo juegados en el sistema

Borrar juego

- **Descripción**: Cualquier usuario logueado puede borrar uno de los juegos creados por él
- **Parámetros de entrada**: Juego a borrar
- **Salida**: En el sistema ya no existe el juego borrado

Abandonar juego

- **Descripción**: Cualquier usuario logueado puede abandonar uno de los juegos creados por él que no esté acabado.
- **Parámetros de entrada**: Juego a abandonar
- **Salida**: El sistema reporta el juego como abandonado

Ver juego

- **Descripción**: Cualquier usuario logueado puede ver el estado de un juego
- **Parámetros de entrada**: Juego a visualizar
- **Salida**: El sistema muestra los datos del juego seleccionado. Éstos incluyen: precio actual de las acciones, cantidad de acciones poseidas por los juegadores, dinero poseido por los jugadores, cartas de influencia y objetivo del jugador.

Jugar turno

- **Descripción**: Cualquier jugador puede realizar acciones en un juego
- **Parámetros de entrada**: Acción a realizar (descritas en los siguientes casos de uso)
- **Salida**: Si la acción ha sido correcta, el jugador verá el juego actualizado. En caso contrario se le mostrará un mensaje de error.

Jugar carta de influencia

- **Descripción**: Cualquier jugador puede jugar una carta de influencia cuando sea requerido por el sistema
- **Parámetros de entrada**: Carta de influencia a jugar
- **Salida**: Los titulos de accion han sido modificados reflejando el comportamiento descrito por la carta de influencia.

Jugar carta de objetivo

- **Descripción**: Cualquier jugador puede jugar una carta de objetivo cuando sea requerido por el sistema
- **Parámetros de entrada**: Carta objetivo
- **Salida**: Si se han cumplido los objetivos, el jugador verá como su dinero se ve incrementado y recibirá una nueva carta de objetivo.

Pagar banco

- **Descripción**: Cualquier jugador puede pagar al banco cuando sea requerido por el sistema
- **Parámetros de entrada**: Cantidad a pagar (50 o 100)
- **Salida**: El jugador verá una transacción monetaria reflejada como negativa en su cuenta y como positiva en la cuenta del banco.

Iniciar compra/venta

- **Descripción**: Cualquier jugador puede iniciar una compra o venta cuando sea requerido por el sistema
- **Parámetros de entrada**: Cantidad de acciones y tipo de acciones.
- **Salida**: El jugador verá como el otro jugador o el banco le compran o venden la cantidad de acciones seleccionada. En caso de que no haya nadie interesado, no habrá reflejada ninguna transacción.

Responder compra/venta

- **Descripción**: Cualquier jugador puede responder a una compra o venta iniciada por otro jugador
- **Parámetros de entrada**: Cantidad de acciones a comprar o vender
- **Salida**: El jugador verá una modificación en su cantidad de acciones y dinero.

Crear juego / seleccionar IA

- **Descripción**: Cualquier usuario logueado puede crear un juego. El sistema le mostrará los tipos de IA disponibles junto a su dificultad y una breve descripción.
- **Parámetros de entrada**: Tipo de IA contra la que se quiere enfrentar
- **Salida**: Un nuevo juego aparece en la lista de juegos

3.3. Mapa conceptual de los datos

Para poder modelizar lo datos de un juego primero hay que identificar cuales son los elementos involucrados en el mismo. Para ello a continucación hay una lista de los elementos mencionados junto una breve explicación de cuales son sus funciones dentro del sistema.

- **Jugador**: El jugador es cualquier usuario que se registre en la página web. Para poder modelizar este elemento necesitaré los siguientes atributos
  - Nombre
  - Email

  - Contraseña
  - Estadísticas del jugador

  Cabe destacar que para la imagen de perfil se ha usado el servicio externo *Gravatar* [4] y por tanto no se necesita ningún atributo ya que con el email del usuario se puede obtener su foto de perfil.

- **Juego**: El juego es el elemento con más importancia dentro de la implementación. Contiene toda la información de una partida.
  - Jugador que lo crea
  - Banco: Aquí se incluye el dinero y las acciones poseidas por el banco
  - Jugador IA: Aquí se incluye el dinero y las acciones poseidas por el sistema
  - Precio de las acciones
  - Estado del juego: los posibles valores son *Abierto, acabado, abandonado, jugando*
  - Ronda actual
  - Número de rondas
  - Ganador (en caso de que ya se haya acabado)
  - Montón de cartas de influencia sin jugar
  - Montón de cartas de objetivo sin jugar
- **Carta de objetivo**: La carta de objetivo ha de modelar los posibles objetivos a cumplir durante el desarrollo del juego. Se ha plasmado fielmente el formato de la carta original
  - Tipo de objetivo: Son los puntos que tiene asignados la carta. Los posibles valores son 450, 500, 550, 600 y 650
  - Parejas de accion y dinero. Estás parejas simbolizan cuanto dinero has de poseer de determinadas acciones para poder cumplir el objetivo
- **Carta de influencia**: Las cartas de influencia, como las cartas de objetivo, han sido fielmente un reflejo de las cartas reales del juego. Hay de dos tipos:
  - Cartas de precio de mercado: son las cartas que hacen que la cotización de una acción pueda modificarse
    - Acción sobre la que realizar una modificación de precio
    - Tipo de modificación: Subir o bajar
    - Cantidad a modificar: Un valor de 1 a 4
  - Cartas de dividendo: son las cartas que proporcionan ciertos beneficios a los poseedores de títulos de acción. Sus atributos son:
    - Tipo de la acción sobre la cual se repartirán dividendos
  - **Turno**: Contiene todas las posibles acciones a realizar durante un turno. En la etapa de diseño se mostrará un diagrama de la transición de estados de un turno. Sus atributos principales son:

- Cantidad pagada al banco
- Cartas de influencia usadas por el jugador
- Carta de objetivo (si se ha usado una)
- Órdenes de compra con sus respuestas
- Órdenes de venta con sus respuestas

A falta de añadir los atributos específicos de la implementación, una posible representación de los modelos sería la siguiente:

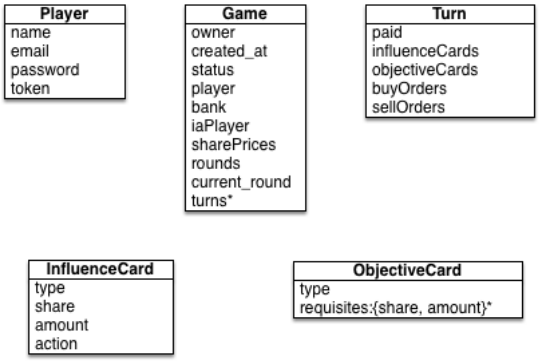

Figura 3.2: Mapa de modelos conceptuales

Cabe destacar que se ha usado la tecnología NoSQL y por tanto no se han establecido las relaciones entre los modelos. EL diseño final de la implementación no es exactamente el mostrado en el diagrama, pero a pesar de ello, se ha mostrado una conceptualización básica para ayudar al lector a plasmar mentalmente lo que se ha querido implementar.

4 Diseño e implementación

En este capítulo se hará especial énfasis en el diseño y la implementación del juego en su versión online. Se dará especial importancia a todas las decisiones que se han tomado a la hora de desarrollar.

4.1. Wireframing y prototipo

Para empezar con el diseño se han realizado unos wireframes (esbozos) de cómo podría ser la interfaz visual del juego para así identificar los elementos que habría que programar.

Se realizaron los siguientes wireframes:

- Pantalla principal

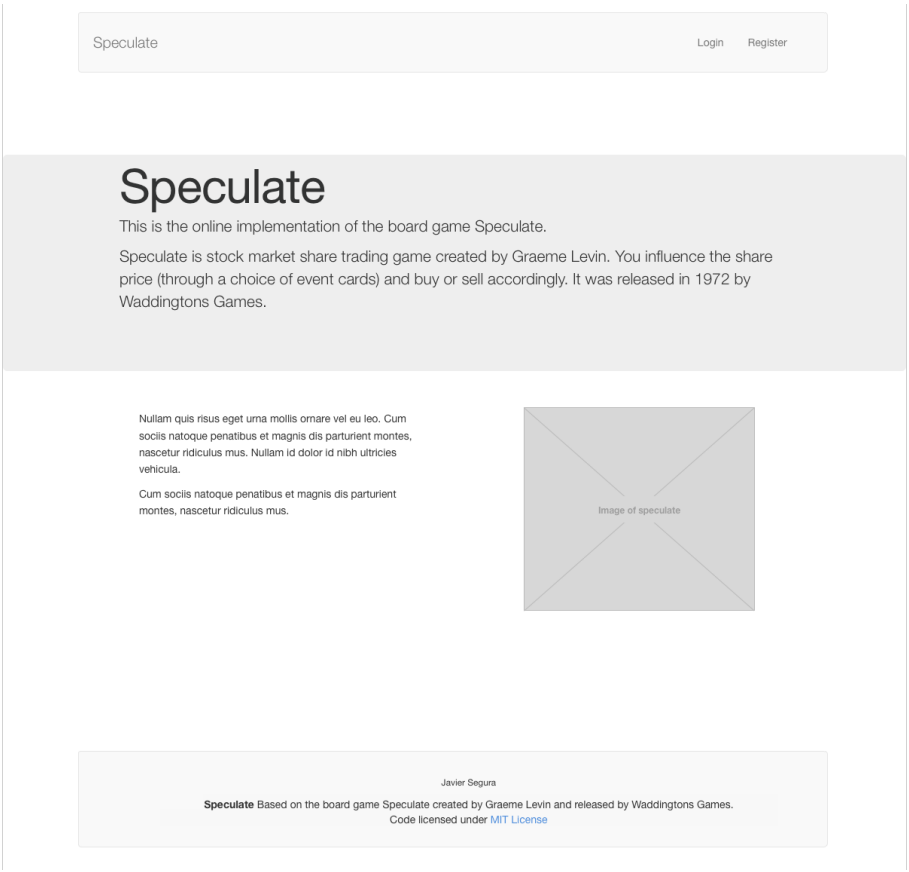
Speculate
Login
Register
Speculate
This is the online implementation of the board game Speculate.
Speculate is stock market share trading game created by Graeme Levin. You influence the share price (through a choice of event cards) and buy or sell accordingly. It was released in 1972 by Waddingtons Games.
Nullam quis risus eget urna mollis ornare vel eu leo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nullam id dolor id nibh ultricies vehicula.
Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.
Image of speculate
Javier Segura
Speculate Based on the board game Speculate created by Graeme Levin and released by Waddingtons Games.
Code licensed under MIT License

Figura 4.1: Wireframe 1

- Login

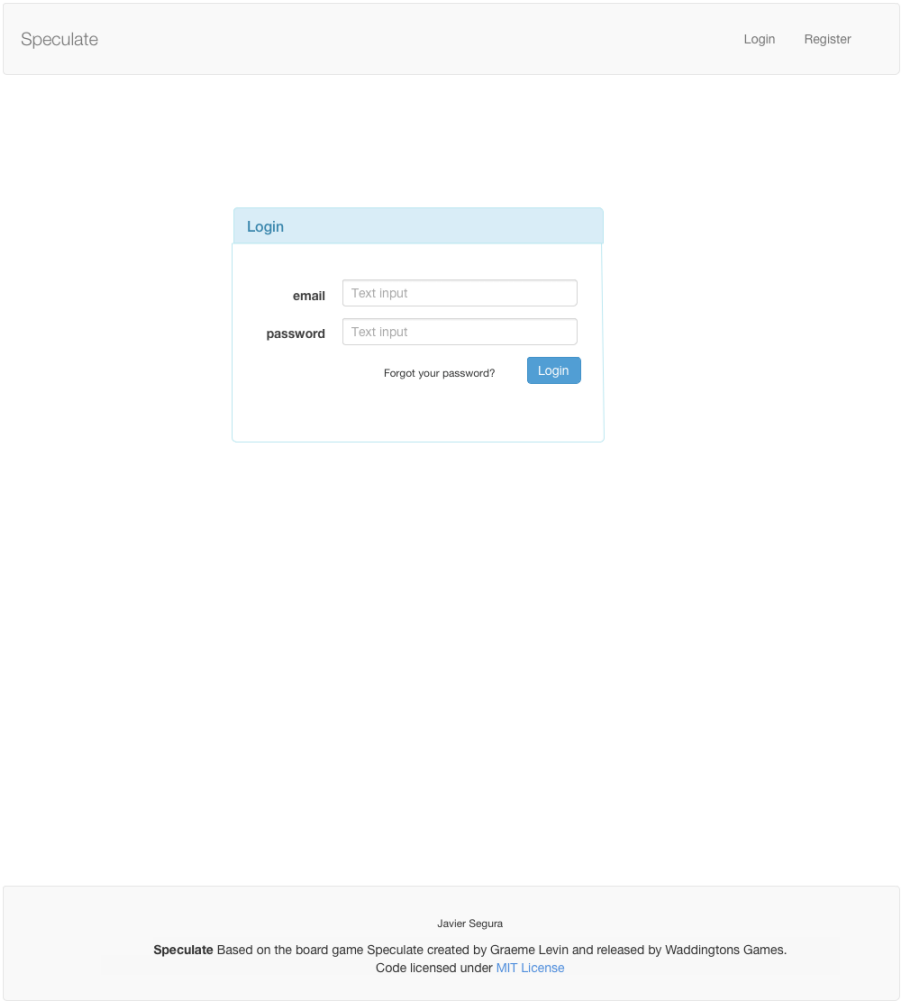

Figura 4.2: Wireframe 2

- Lista de juegos

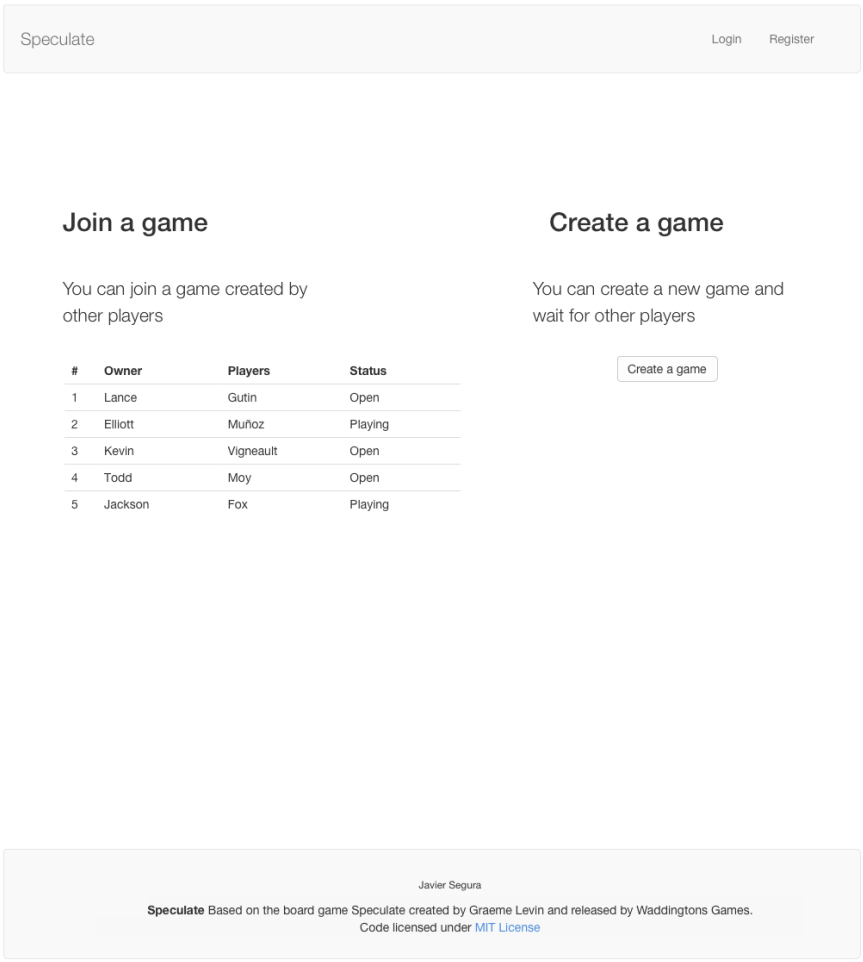
Speculate
Login
Register
Join a game
You can join a game created by other players
Owner Players Status
1 Lance Gutin Open
2 Elliott Muñoz Playing
3 Kevin Vigneault Open
4 Todd Moy Open
5 Jackson Fox Playing
Create a game
You can create a new game and wait for other players
Create a game
Javier Segura
Speculate Based on the board game Speculate created by Graeme Levin and released by Waddingtons Games.
Code licensed under MIT License

Figura 4.3: Wireframe 3

- Vista de un juego

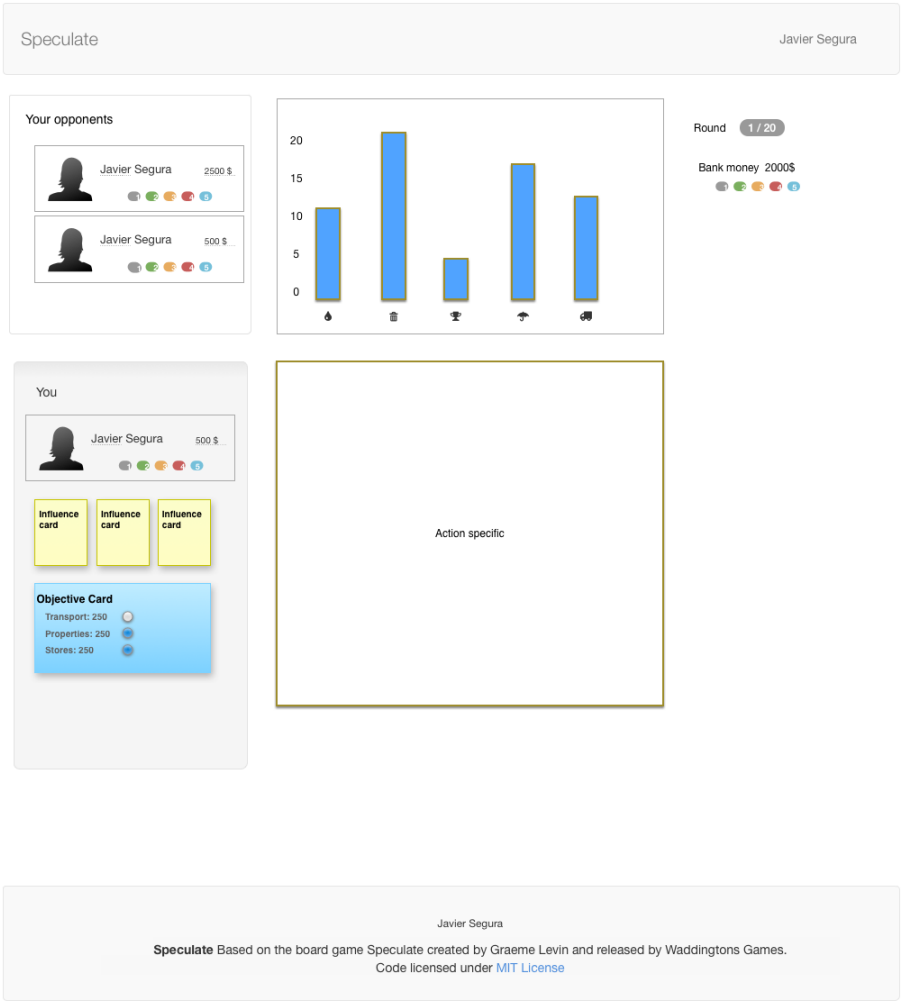

Figura 4.4: Wireframe 4

4.1.1. Resultado final

Para ilustrar el cambio sufrido por el wireframing a través de todas las iteraciones desarrolladas, adjunto las imágenes sacadas del diseño final del juego:

- Pantalla de login

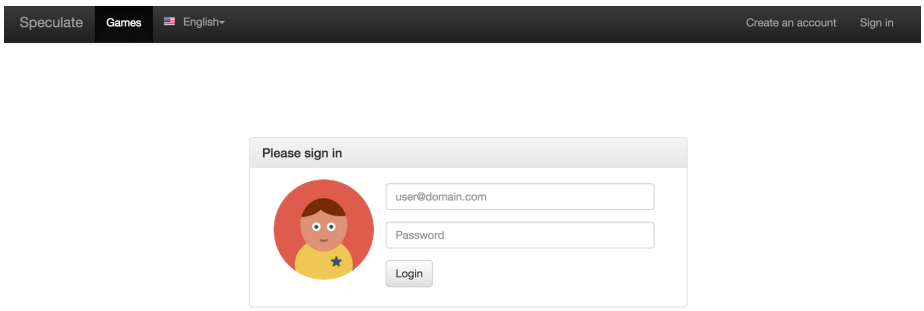

Figura 4.5: Pantalla de login

- Pantalla principal

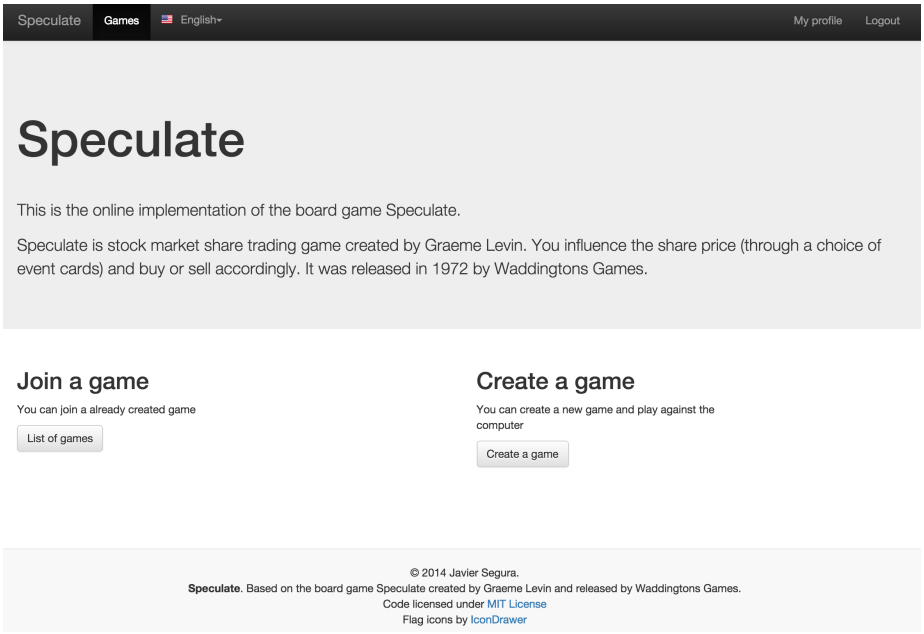
Speculate
Games
English
My profile
Logout
Speculate
This is the online implementation of the board game Speculate.
Speculate is stock market share trading game created by Graeme Levin. You influence the share price (through a choice of event cards) and buy or sell accordingly. It was released in 1972 by Waddingtons Games.
Join a game
You can join a already created game
List of games
Create a game
You can create a new game and play against the computer
Create a game
© 2014 Javier Segura.
Speculate. Based on the board game Speculate created by Graeme Levin and released by Waddingtons Games.
Code licensed under MIT License
Flag icons by IconDrawer

Figura 4.6: Pantalla principal

- Registro

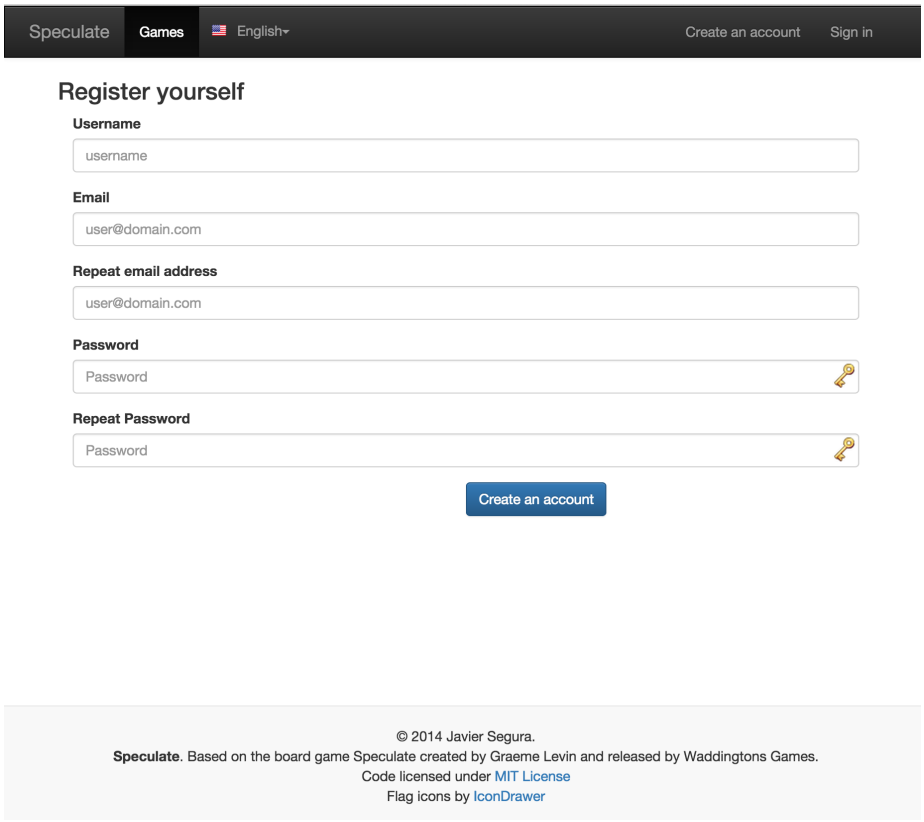

Figura 4.7: Registro

- Perfil

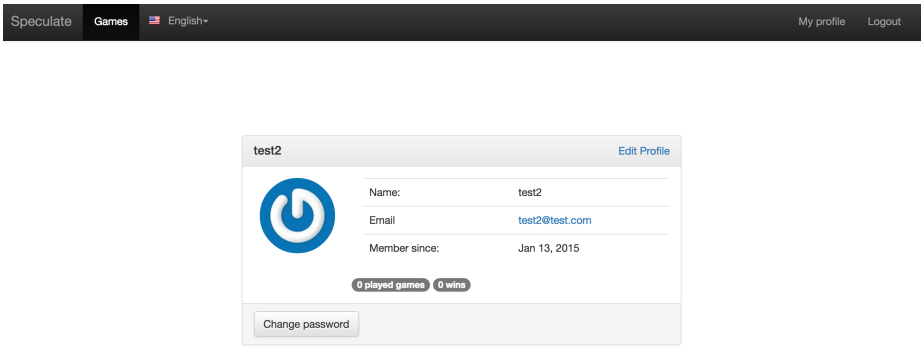

Figura 4.8: Perfil

- Lista de juegos

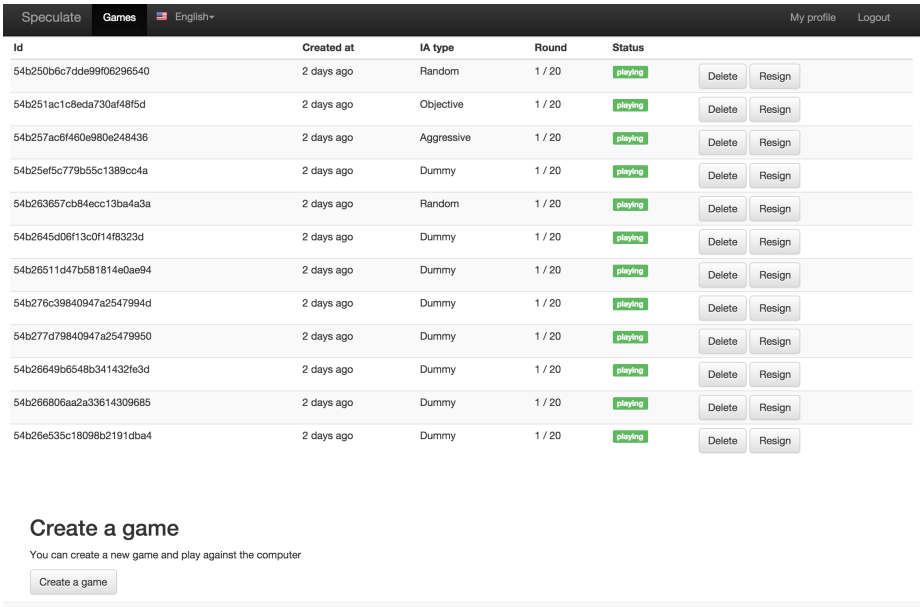
Speculate
Games
English
My profile
Logout
Id
Created at
IA type
Round
Status
54b250b6c7dde99f06296540 2 days ago Random 1 / 20 playing Delete Resign
54b251ac1c8eda730af48f5d 2 days ago Objective 1 / 20 playing Delete Resign
54b257ac6f460e980e248436 2 days ago Aggressive 1 / 20 playing Delete Resign
54b25ef5c779b55c1389cc4a 2 days ago Dummy 1 / 20 playing Delete Resign
54b263657cb84ecc13ba4a3a 2 days ago Random 1 / 20 playing Delete Resign
54b2645d06f13c0f14f8323d 2 days ago Dummy 1 / 20 playing Delete Resign
54b26511d47b581814e0ae94 2 days ago Dummy 1 / 20 playing Delete Resign
54b276c39840947a2547994d 2 days ago Dummy 1 / 20 playing Delete Resign
54b277d79840947a25479950 2 days ago Dummy 1 / 20 playing Delete Resign
54b26649b6548b341432fe3d 2 days ago Dummy 1 / 20 playing Delete Resign
54b266806aa2a33614309685 2 days ago Dummy 1 / 20 playing Delete Resign
54b26e535c18098b2191dba4 2 days ago Dummy 1 / 20 playing Delete Resign
Create a game
You can create a new game and play against the computer
Create a game

Figura 4.9: Lista de juegos

- Vista de juego. Acción pagar

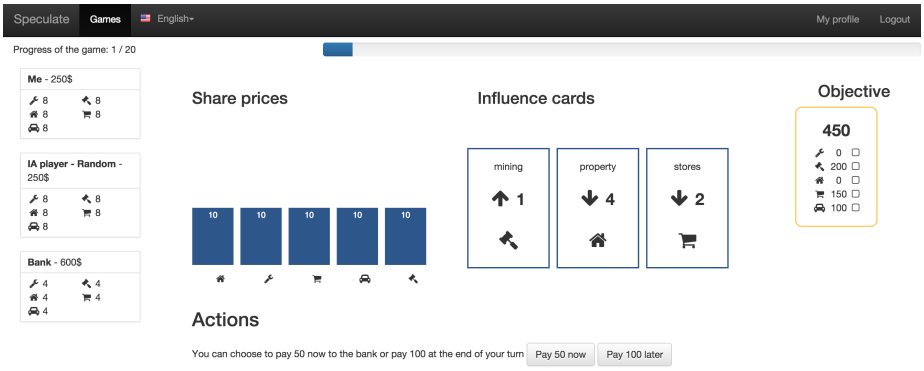

Figura 4.10: Vista de juego. Acción pagar

- Vista de juego. Acción jugar carta de influencia

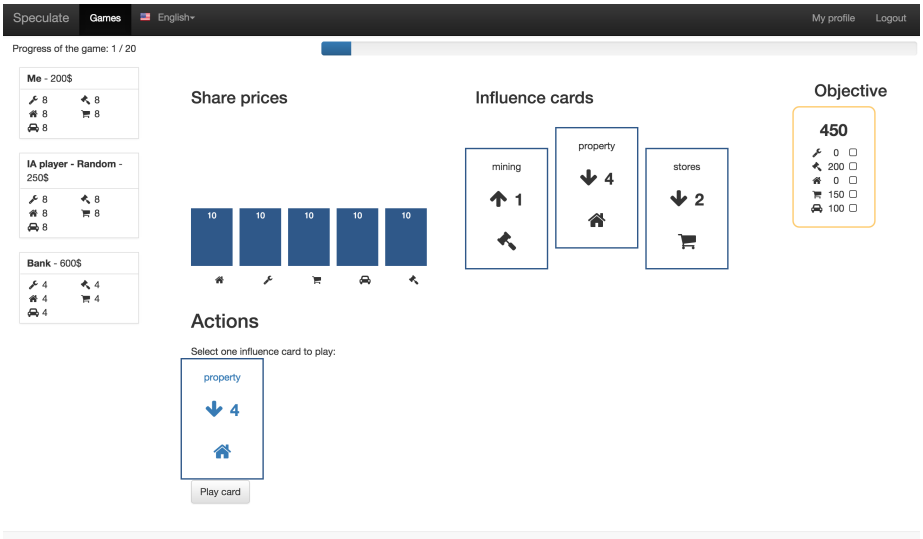

Figura 4.11: Vista de juego. Acción jugar carta de influencia

- Detalle de compra y venta de acciones

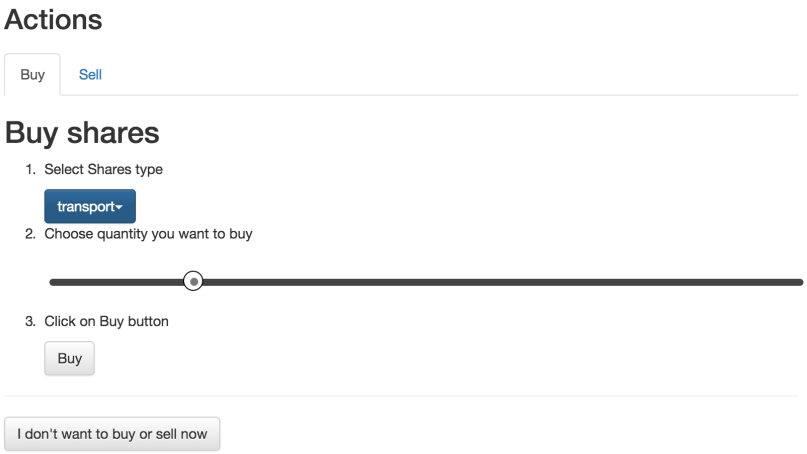

Figura 4.12: Detalle de compra y venta de acciones

4.2. Entorno y plataforma

Como ya se ha comentado, uno de los problemas que hay que tener en cuenta en cualquier implementación web usando tecnologías actuales es la escalabilidad.

Para este problema hay una gran variedad de soluciones, en mi implementación he optado por un módelo de 3 capas: la capa de cliente (con cierta lógica), un servidor sin estado que contiene la mayoría de la lógica de aplicación y la capa de persistencia de datos. En el desarrollo de todas ellas se han seleccionado tecnologías que permitan escalar horizontalmente sin problemas.

4.2.1. Cliente web

Para el desarrollo del cliente web se ha optado por utilizar una SPA (Single Page Application). Para ello se ha hecho uso de AngularJS, un framework publicado por Google que ha recibido una gran aceptación por parte de la comunidad. Es un proyecto de código abierto, realizado en Javascript que contiene un conjunto de librerías útiles para el desarrollo de aplicaciones web y propone una serie de patrones de diseño para llevarlas a cabo. En pocas palabras, es lo que se conoce como un framework para el desarrollo, en este caso sobre el lenguaje Javascript con programación del lado del cliente.

El framework usa el paradigma MVC (Modelo, Vista y Controlador) cuyos componentes principales son los siguientes:

- Vistas: Será el HTML y todo lo que represente datos o información.

- Controladores: Se encargarán de la lógica de la aplicación y sobre todo de las llamadas “Factoría” y “Servicios” para mover datos contra servidores o memoria local en HTML5. - Modelo de la vista:

- Modelo: En Angular el "Modelo" es algo más de aquello que se entiende habitualmente cuando te hablan del MVC tradicional, osea, las vistas son algo más que el modelo de datos. En modo de ejemplo, en aplicaciones de negocio donde tienes que manejar la contabilidad de una empresa, el modelo serían los movimientos contables. Pero en una pantalla concreta de tu aplicación es posible que tengas que ver otras cosas, además del movimiento contable, como el nombre de los usuarios, los permisos que tienen, si pueden ver los datos, editarlos, etc. Toda esa información, que es útil para el programador pero que no forma parte del modelo del negocio, es a lo que llamamos el "Scope" que es el modelo en Angular.

Además de todo lo descrito, podemos modularizar la aplicación en lo que denominaremos "Módulos". Se ha hecho un uso extensivo de los módulos en la programación de frontend. Eso nos proporciona el poder de aislar cada uno de los componentes.

4.2.2. Servidor

En la capa de servidor se ha escogido Node.js. Node.js es un entorno de programación en la capa del servidor basado en el lenguaje de programación ECMAScript, asíncrono, con I/O de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google. En su creación se tuvo en cuenta el enfoque de crear infraestructuras altamente escalables.

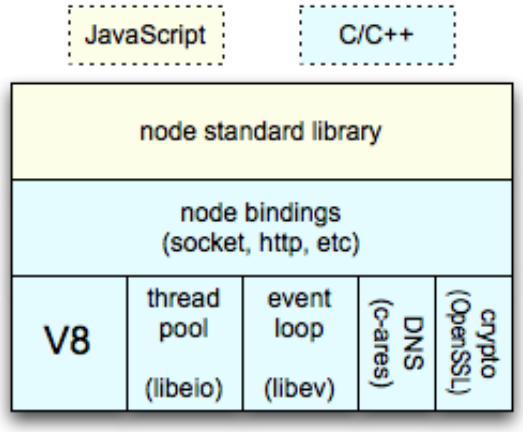

Figura 4.13: Arquitectura de Node.js

Una de las ventajas de escojer Node.js es que utlizaremos la tecnología Javascript tanto en el servidor como en el cliente, cosa que nos facilitará el desarrollo y nos permitirá hacer entregas de código más rápidamente; no tenemos que utilizar diferentes lenguajes de programación.

4.2.3. Capa de datos

Para la persistencia de datos se han valorado diferentes alternativas:

MongoDB: es una base de datos NoSQL orientada a documentos. Dos de sus grandes ventajas es que es "schemaless" (no hay esquema fijo de los datos) y

la otra es que el formato para interactuar se parece mucho a JSON, (MongoDB lo llama BSON), y eso nos facilita el intercambio de datos entre la capa de servidor y la capa web.

Al final la elección se ha decantado por MongoDB por los siguientes factores:

- Escalabilidad
- Rapidez
- Robustez
- Facilidad de uso

4.2.4. Librerías utilizadas

Aquí listo las librerías más importantes que se han utilizado para poder llevar a cabo la implemtación de Speculate.

4.2.5. Frontend

- Lo-dash [5] es una librería que proporciona una gran cantidad de funciones para trabajar con programación funcional las cuales son compatibles con todos los intérpretes de Javascript (navegadores y Nodejs). La elección de ésta frente a Underscore ha sido debida a que su tamaño es mucho menor y requiere menos dependencias. Se ha utilizado tanto en backend como en frontend, ya que como se ha mencionado anteriormente, ambas partes se han desarrollado con Javascript.
- Bootstrap [6] es un framework de frontend que proporciona una base de hojas de estilo CSS para proyectos web. También provee componentes ya diseñados que ayudan a mejorar el estilo. La elección ha sido debida a la falta de conocimientos en diseño y a su fácil aprendizaje.
- Grunt [7] es un gestor de tareas en Javascript (parecido a la herramienta make de GNU). Ha sido utilizada para la compilación y ejecución de los ficheros servidos por backend. Se han configurado las siguientes tareas:
  - *serve* Lanza el servidor de backend en modo desarrollo. Dicho modo relanza el servidor ante cualquier cambio en los ficheros, cosa que nos facilita y agilita el desarrollo. También hace que los clientes conectados mediante la web refresquen el contenido ya que les envía la petición de actualización via websockets.
  - *localize* Ayuda a completar todos los ficheros de internacionalización del juego. Ésto nos hace que ningún texto quede sin traducir.

4.2.6. Backend

- Mongoose [8] es una librería para mapear objetos a modelos de mongodb en Node.js. Mongoose provee una interfaz casi directa usando esquemas para modelar todos los datos de tu aplicación. También incluye gestión de tipos de datos, transformaciones, validaciones, lógica de consulta y todo ello sin tener que desarrollar módulos específicos para ello. Toda la comunicación con MongoDB se ha hecho a través de esta librería.

4.3. Estados de un juego

Para familiarizar al lector sobre la estructura se ha creado un gráfico autoexplicativo de las posibles transacciones y cambios de estado que se realizan durante la duración de un juego. Si se desean más detalles al respecto se puede consultar el manual de usuario que se ha traducido especialmente para la redacción de esta memoria

Jugador inicia la partida
Empieza su turno
Paga 50 al banco
Juega primera carta de influencia
Inicia una venta de acciones
Los jugadores responden
Juega segunda carta de influencia
Inicia una compra de acciones
Juega carta de objetivo
Finaliza el juego - Se declara un ganador
Responde a una venta de acciones
Paga 100 si no ha pagado antes 50
Espera mientras otros jugadores juegan su turno
Responde a una compra de acciones

Figura 4.14: Diagrama de secuencia y estados

4.4. Implementación de la IA

Uno de los requisitos del juego era poder jugar contra un jugador ficticio. Para ellos se han implementado en la capa de servidor 4 tipos de jugadores ficticios, cada uno con una personalidad diferente.

- **Dummy**: Este tipo de IA es muy simple, básicamente lo único que hace es jugar una carta aleatoria de influencia y no compra o vende acciones durante la duración de la partida. Eso hace que en la mayoría de casos, pierda las partidas.
- **Aleatorio**: Este tipo de IA realiza acciones aleatorias. Reliza alguna compra y venta de un tipo aleatorio de acciones. No tiene en cuenta los objetivos a cumplir.
- **Objetivo**: Este tipo de IA intenta cumplir los objetivos definidos en la carta de objetivo. La principal diferencia con el jugador aleatorio es que las compras o ventas de acciones que realiza intentan seguir los objetivos

marcados en su carta de objetivo, eso hace que aparte de los beneficios de esas actuaciones, vaya consiguiendo dinero a través de los objetivos cumplidos.

- **Agresivo**: Este tipo de IA intenta maximizar los beneficios cuando realiza acciones. No tiene en cuenta las cartas de objetivo pero realiza las compras o ventas teniendo en cuenta el beneficio que puede obtener. Básicamente intenta plasmar el comportamiento que he observado en la mayoría de jugadores contra los que me he enfrentado

4.5. Implementación REST de backend

Para poder satisfacer el objetivo de la extensibilidad, se ha implementado el backend siguiendo el estándard REST, que nos proporciona los siguientes beneficios:

- Protocolo cliente-servidor sin estado. Ésto hace que todos los mensajes contengan toda la información necesaria y evita que tanto el cliente como el servidor tengan que almacenar el estado de las peticiones.
- Conjunto de operaciones definidas. Se utilizan los verbos de HTTP: *GET, POST, DELETE, PUT*. Es un idioma universal que cualquier aplicación, ya sea nativa o web, puede utilizar. De esta manera nos aseguramos que futuros desarrollos puedan utilizar la tecnología de backend que se ha implementado.

De esta manera se han programado las siguientes rutas dentro del servidor para poder manejar las peticiones iniciadas por el cliente web:

- PUT /login: Enviar información de login. Se devuelve el token a incluir en el resto de peticiones a partir de ahora.
- PUT /logout: Borrar el token de usuario
- GET /me: Recibir información de usuario
- GET /api/users/: Obtener la lista de usuarios
- GET /api/users/ID: Obtener el usuario identificado por ID
- GET /api/games: Obtener la lista de juegos
- GET /api/games/ID: Obtener el juego identificado por ID
- POST /api/games: Crear un juego nuevo
- POST /api/games/ID/resign: Abandonar un juego
- DELETE /api/games/ID: Borrar el juego identificado por ID.

Con la implementación de un cliente para estas rutas HTTP y con la comunicación en tiempo real (explicada en este capítulo), cualquier desarrollador podría un nuevo interfaz sin tener que implementar de nuevo todo el backend del juego. Incluso esto podría ser un futuro proyecto de final de carrera.

4.6. Autenticación de los clientes

Para aseguarse de que las peticiones son hechas por un usuario autenticado se ha hecho uso de la tecnología *JSON Web Token* [9].

La implementación consta en dos fases.

En la primera fase, cuando el usuario decide loguearse en el sistema, se calcula un token que se le devuelve al usuario y a su vez se guarda en base de datos.

La segunda fase comprende todas las comunicaciones entre el cliente y el servidor. En esa fase, cualquier petición HTTP que se realice desde el cliente, incluye una cabecera HTTP del tipo (Donde TOKEN es el token generado por el sistema):

```
Authorization: Bearer TOKEN
```

Una vez recibida la petición del cliente, el servidor de backend comprueba si el token es un token válido y así consigue saber de qué usuario es la petición que ha recibido.

4.7. Implementación de la comunicación en tiempo real

Uno de los objetivos identificados en el proyecto era ponerse al día en lo que a desarrollo web se refiere. Hoy en día, y cada vez más, la comunicación en tiempo real es un factor clave que puede llegar a decidir el éxito de cualquier aplicación.

Se ha seguido esta filosofía dentro de la implementación de *Speculate* mediante el uso de la tecnología Websockets. Tal y como lo define la Wikipedia: *WebSocket es una tecnología que proporciona un canal de comunicación bidireccional y full-duplex sobre un único socket TCP*

Los Websockets es una tecnología bastante moderna que está implementada en todos los navegadores modernos.

4.7.1. Uso dentro de Speculate

En la implementación del juego se han usado los Websockets para realizar toda la comunicación que hay dentro de un juego. El enfoque que se ha seguido ha sido bastante práctico, he usado mensajes *JSON*[10] (un formato de intercambio de datos bastante ligero), para enviar las posibles acciones que se realizan mientras el usuario interactua con una partida. Eso hace que todos los usuarios que tengan una partida abierta, estarán recibiendo esos mensajes, y por tanto mantengan actualizado el estado del juego.

A continuación se listan los ejemplos de mensajes que se han utilizado para la comunicación (corresponden a todos los casos de uso posibles dentro de la partida). Todos ellos tienen ciertos campos comunes, necesarios para identificar el contexto de la acción que se realiza.

- **Pagar al banco**: El único parámetro especial es la cantidad que se paga al banco.

```
{
  "action": "pay",
  "game_id": "54b250b6c7dde99f06296541",
  "player_id": "54a0511d4bf5c722d13d7404",
  "amount": 50
}
```

- **Jugar carta de influencia**: El parámetro especial es la carta a ser jugada.

```
{
"action": "playInfluenceCard",
"game_id": "54b250b6c7dde99f06296541",
"player_id": "54a0511d4bf5c722d13d7404",
"idcard": "mining1up"
}
```

- **Jugar carta de objetivo**: El parámetro especial es la carta a ser jugada.

```
{
"action": "playObjectiveCard",
"game_id": "54b250b6c7dde99f06296541",
"player_id": "54a0511d4bf5c722d13d7404",
"idcard": "mst450"
}
```

- **Iniciar compra o venta de títulos de acción**: El parámetro especial es el título de acción y la cantidad

```
{
"action": "buy",
"game_id": "54b250b6c7dde99f06296541",
"player_id": "54a0511d4bf5c722d13d7404",
"amount": 5,
"share": "mining"
}
```

Todos los mensajes pueden recibir dos mensajes de respuesta del servidor:

- **Acción correcta**: Si la acción es correcta le pide al cliente web que refresque el juego.

```
{
"action": "update",
"game_id": "54b250b6c7dde99f06296541",
}
```

- **Error**: Si la acción es erronea, devuelve error y el código del mismo.

```
{
"action": "error",
"game_id": "54b250b6c7dde99f06296541",
"error_code": "Not owner of the card"
}
```

5 Pruebas

Durante cualquier desarrollo de programación tiene que estar presente la fase de Testing, en este caso no iba a ser diferente. Se han realizado dos tipos de test: test unitarios y tests manuales.

5.1. Tests unitarios

Los tests unitarios son aquellos que comprueban el funcionamiento de un componente aislado en el sistema. En *Speculate* se han realizado tests en Backend que cubren tanto el modelo *Jugador* como el modelo *juego*. Cabe decir que no han sido todo lo extensos que me gustaría, pero eso ha sido debido a la falta de tiempo.

Se ha usado la librería Mocha.js que nos proporciona la integración que necesitamos. Se ha configurado el sistema de puesta en marcha en producción para que se ejecuten los tests antes de poder subir cualquier nueva versión. Eso nos proporciona la seguridad de que no introducimos ningún nuevo error en el código antiguo.

Para ilustrar al lector, se ha printado una de la salida de los tests realizados

```
Authorization
POST /logout 401 1.883 ms - 24
- /logout requires token
GET /api/games 401 0.872 ms - 24
- /api/games requires token
GET /register 404 0.916 ms - 21
- /register doesn't require token
GET /login 404 1.046 ms - 18
- /login doesn't require token

Games
- retrieve a game by owner
- retrieve a game by id
- it's possible to list games
Creating a game
```

```
- should create a new game
- with valid fields
- should update the total game count
Joining a game
- should update the players game count
- should fail if the game is not accepting players
starting a game
- should be ok if there is enough players
- update the game details
- should deal cards, money and shares to the players

InfluenceCards
- has the defined cards
- has the defined price cards
- has the defined dividend cards

ObjectiveCards
- has the defined cards

Users
- retrieve a user by email
- it's possible to list users
Register
- register a new user
- should fail when registering a duplicate email
- should populate the creation data
- should not find a non existing user
- should update the total users count
- should encrypt the password
Login
- should remove the token when a user logout


29 passing (613ms)
```

Lo que se ha intentado hacer con estos tests ha sido verificar las llamadas que tienen los modulos entre sí.

5.2. Tests manuales

Una de las partes importantes del juego, por no decir la que mayor importancia tiene, es la inteligencia artificial de los jugadores.

Debido a problemas con la ejecución del código no he podido realizar tests entre jugadores controlados por el ordenador. Pese a ello, se han realizado partidas contra la máquina en cada uno de los niveles.

En concreto se han seleccionado dos jugadores: **A**, un jugador con experiencia en el juego de mesa *Speculate* y **B**, sin experiencia ninguna, dando lugar a los siguientes resultados mostrados en la tabla.

Se han realizado un total de 40 partidas, 5 por cada jugador y tipo de inteligencia. El porcentaje mostrado es el ratio entre las partidas ganadas y las

partidas jugadas.

	Dummy	**Aleatorio**	**Agresivo**	**Objetivo**
A	80 %	80 %	60 %	40 %
B	100 %	100 %	80 %	80 %

De los datos se pueden extraer las siguientes conclusiones:

1. La dificultad del juego cuando ya se está familiarizado con él es baja. Se tendría que mejorar sobre ese punto.

2. El que un jugador no haga nada o realice acciones aleatorias no difiere demasiado. En ambos casos el jugador está muy por encima de él.

3. Una posible IA sería una mezcla entre la consecución de objetivos y la consecución de beneficios inmediatos. Ya que está visto que si se olvida cualquier de los dos factores no se consigue un rendimiento óptimo.

6 Planificación y estudio económico

En este capítulo se hará un análisis de la planificación temporal del proyecto. Se analizaran las diferentes etapas principales del desarrollo del proyecto, haciendo una comparativa entre el tiempo estimado y el tiempo real dedicado para la ejecución de la implementación. También se analizará el coste económico del proyecto, tanto en recursos humanos como en herramientas utilizadas.

6.1. Planificación

6.1.1. Estimación

Aquí presento una planificación temporal de la realización de este proyecto. Se ha desglosado en varias etapas todas consecutivas ya que la realización sólo ha contando con una persona, aunque haya ejercido diversos roles dentro del proyecto. Se estima que la duración del proyecto es de 681h.

Figura 6.1: Planificación

El desglose de las horas planificadas por tareas es el siguiente:

Tarea	Horas planificadas
Estudio del juego SPECULATE	80h
Análisis general de requisitos	16h
Evaluación de tecnologías	50h
Preparación del entorno de trabajo	30h
Prototipo & wireframing	40h
Backend	90h
Frontend	150h
UI	70h
Testing	80h
Redacción de la memoria	75h
Total	681h

6.1.2. Realidad

Después de haber acabado el desarrollo del proyecto he descubierto que la estimación y la realidad no concuerdan. Aquí se muestra la diferencia entre ambas:

Tarea	Horas planificadas	Horas invertidas
Estudio del juego SPECULATE	80h	45h
Análisis general de requisitos	16h	16h
Evaluación de tecnologías	50h	61h
Preparación del entorno de trabajo	30h	35h
Prototipo & wireframing	40h	74h
Backend	90h	112h
Frontend	150h	132h
UI	70h	97h
Testing	80h	79h
Redacción de la memoria	75h	80h
Total	681h	731h

6.1.3. Desviación de la planificación inicial

Como podemos observar, durante del desarrollo se ha invertido más tiempo del que estaba planificado en las tareas excepto en la tarea de Frontend. A pesar de ésto no se han llegado a cumplir todos los objetivos que se habían previsto, siendo el más importante el soporte para multijugador. Ahora, con más conocimiento sobre las herramientas utilizadas puedo decir que la adaptación del juego para poder aceptar a varios jugadores por partida llevaría 3 semanas.

Uno de los principales motivos de la desviación en la planificación ha sido la propia planificación, es decir, al no tener suficientes conocimientos sobre la arquitectura y el desarrollo en Node.js, los tiempos no están correctamente estimados. Se ha arreglado reduciendo el conjunto de funcionalidades que se entregan en este proyecto, cosa que en un entorno profesional no debería pasar.

6.2. Análisis económico

Para realizar un análisis económico se han tenido en cuenta dos factores:

- **Horas invertidas**. Dando un total de 32635 €.
- **Coste equipamiento**. Dando un total de 3185 €.

El coste total del proyecto asciende a 35820 €.

6.2.1. Horas invertidas

Una vez plasmada la planificación real podemos separar las etapas en función del nivel laboral que éstas requieren. Separemos las horas en función de si la persona que ha de realizar el trabajo es un Analista, un Programador o un Jefe de proyecto. Para poder calcular los costes se han definido los siguientes precios por hora de cada uno de los cargos:

Cargo	Precio por hora
Jefe de proyecto	61 €
Analista	45 €
Programador	35 €

Cálculo del coste total de los roles involucrados:

Cargo	Horas	Total
Jefe de proyecto	175h	10675 €
Analista	250h	11250 €
Programador	306h	10710 €
Coste total		32635

El coste total en personal en el proyecto es de 32635 €.

6.2.2. Coste de equipamiento

En la siguiente tabla se puede ver un desglose de todos las herramientas o servicios externos que se han necesitado para la ejecución del proyecto.

Concepto	Precio mensual	Meses	Precio
Servidor Mongodb	12 €	6	72 €
Servidor Nodejs	28 €	6	168 €
Cuenta privada GitHub (repositorio código)	6 €	6	36 €
Subscripción egghead.io (tutorial angularjs)	14 €	6	84 €
Alquiler oficina en espacio compartido	250 €	6	1500 €
Ordenador personal	1200 €		1200 €
Material oficina	15 €	6	90 €
Juego de mesa original	35 €		35 €
Total			3185 €

7 Conclusiones

El objetivo del proyecto era crear un juego multijugador basándose en un juego de mesa publicado en el año 1970. Aunque el objetivo no se ha cumplido, debido a como se ha hecho la comunicación de los jugadores con el servidor, adaptarlo a multijugador sería un trabajo fácilmente realizable en 3 semanas.

Los motivos que han hecho que no se cumpla han sido:

- Falta de conocimiento de las aplicación Single Page y el aprendizaje del framework utilizado.
- Falta de conocimiento en la gestión de bases de datos en Mongodb.
- Poca experiencia en el diseño de páginas web adaptables a varios navegadores.

Ésto puede ser una base para futuros proyectos y me gustaría hacer algunas sugerencias sobre cosas que mi implementación de Speculate puede mejorar:

- Mejor feedback al jugador: Actualmente el feedback retornado al jugador no es todo lo preciso que cabe esperar de un juego online.
- Historial del juego para poder recrear las partidas. Se puede crear un historial de acciones realizadas las cuales puedan ser ejecutadas periódicamente para que un usuario pueda visualizar partidas antiguas. Ésto junto a un buen feedback gráfico podría usarse como herramienta de aprendizaje para posibles nuevos jugadores.
- Aprendizaje automático en la IA. Actualmente las IA programadas no tienen aprendizaje sobre las acciones que realizan. Una posible mejora del proyecto sería añadir un motor de aprendizaje para poder jugar contra jugadores controlados por la máquina que cada vez sean más difíciles.
- Infraestructura replicada geográficamente.

También me gustaría felicitar a Miquel Barceló por sus propuestas de proyectos, son temas bastante más atrayentes que la mayoría de propuestas de proyectos de otros profesores. Después de varios años teniendo el proyecto de final de carrera pendiente, las ofertas propuestas por Miquel son uno de los factores que me ha llevado a realizarlo. Gracias Miquel.

Apéndices

A

Glosario

datacenter Centros de procesamiento de datos. 13

git Control de versiones Git. 70

NoSQL Bases de datoos no relacionales. 14

REST Transferencia de Estado Representacional. 39

SPA Single Page Application. 35

Websockets Comunicaciones en tiempo real y en ambos sentidos mediante un solo canal TCP. 14, 40

B

Bibliografía

[1] "Speculate in boardgeekgame." http://boardgamegeek.com/boardgame/2104/speculate, Febrero 2014.

[2] "Internet usage statistics." http://www.internetworldstats.com/stats.htm, Enero 2015.

[3] D. Capital, "La industria del videojuego valdrá más de 100.000 millones de dólares en 2017.." http://goo.gl/1PFUbK, Enero 2014.

[4] "Gravatar: a global recognized avatar." https://en.gravatar.com/, 2014.

[5] "Lo-Dash." http://lodash.com/, Enero 2015.

[6] "Bootstrap." http://getbootstrap.com/, Enero 2015.

[7] "Grunt: The javascript task runner." http://gruntjs.com/, Enero 2015.

[8] "Mongoose library." http://mongoosejs.com/, Enero 2015.

[9] "Json web token." http://jwt.io/, 2014.

[10] "Json." http://json.org/, 2014.

[11] J. Cravens and T. Q. Brady, *Building Web Apps with Ember.js.* Write ambitious Javascript, O'Reilly, 2014.

[12] G. V. Feruglio, *Composición de textos científicos con LaTeX* . Edicions UPC, 1997.

[13] A. Andersson, "Multiplayer game server for turn-based mobile games in erlang," Master's thesis.

[14] "Licencia MIT." http://opensource.org/licenses/MIT, Noviembre 2014.

[15] J. Segura, "Speculate." https://github.com/jsegura/speculate, Enero 2015.

[16] B. G. . S. Seshadri, *AngularJS.* O'Reilly, 2013.

[17] M. Inc, "Mongo." http://www.mongodb.org/.

[18] Joyent, "Node.js." http://nodejs.org/.

C

Manual del juego original

El juego original estaba distribuido con un manual que se incluye en el documento. La versión en castellano ha sido traducida especialmente para incluirla en esta memoria de proyecto.

C.1. Manual original

C.1.1. The aim of the game

This is a game based firmly on the characteristics of dealings in stocks and shares. As in reality, the strategies of the investors decisively affect the market trends —with the chance element of only secondary importance.

Speculate does not provide a situation in which players win or lose unrealistic millions, it simulates the position of the small investor trying to increase the value of his holdings by the modest but vital amounts he knows in real life he can hope for. Now, however, he is in competition with others, each of whom will try to turn the market situation in his own favour.

There will be times when players will have the excitement of seeing their plans bring substantial gains but if the market does not run their way they will find an equally exciting and challenging situation as they struggle to hold their portfolios together and avoid the slide into bankruptcy!

C.1.2. Strategy

Players should be aware that they themselves can radically affect the market, producing bull or bear conditions. There will inevitably be periods of market depression as well as times when it is bouyant. Some players are tempted, after they have completed their trading, always to depress the market if they can in order to lower the value of their opponents' holdings. This is at times a perfectly proper strategy to adopt but it is never wise to make an absolute habit of it!

Some players may be tempted to wait until shares have fallen to very low levels before making their bids, in the hope of buying cheaply. They should be aware that in this situation other players will be unwilling to sell and the market is likely to become increasingly stagnant.

During the game, players are given "objectives" —of portfolios of certain shares —to achieve. When they succeed in reaching their objectives, players gain a useful cash bonus. However, players should not concentrate on achieving their objectives to such an extent that they fail to take advantage of the many trading opportunities that will present themselves during the course of play. There may well be times when it will pay to abandon temporarily the effort to achieve the "objective".

C.1.3. Equipment

1. 2750 in banknotes (this total is deliberately set in order to keep money "tight")
2. Pack of Market influence cards (including 5 Dividend cards)
3. Objective cards (10 of each of 5 different values)
4. 50 Shares of each of 5 types (Certificates in units of 1, 2 &3) (Mining, Stores, Property, Transport and Engineering)
5. Market Price Index (in the box lid)
6. 5 markers for the Price Index board

The tray in the box is used in a way that represents the general stock market. It should be thought of as more than simply a Bank — but "Bank" is a convenient word to use and it is therefore used in these instructions.

C.1.4. Preparation

1. Appoint a Banker to control dealings with the Bank and another player to control the Price Index board.
2. Empty the tray of all its contents.

   Separate the two sections of the calculators and fold over the top ends of the selector strips. These strips will then slide across the columns of figures.
3. Place the five markers on the Market Index board to show a price of 10 for each type of share.
4. Give each player 250 in cash and 8 shares of each type — i.e. — 40 shares in total.
5. Return to the Bank

   a) 300 in cash multiplied by the number of players.

   b) 2 shares of each type multiplied by the number of players.

   For example — if three are playing, the Bank will have 6 shares of each type and 900 in cash.

   All other Shares and banknotes must be put on one side — they are not used in the game. If 5 are playing, all the Shares and banknotes will, of course, be used.

6. Separate the Objective cards into the five different values. Shuffle each set and place them face down, in ascending order, in the tray. Deal one 450 Objective card to each player.

7. Shuffle the Market Influence cards. Deal three to each player and place the remainder face down in the tray.

   Players must not be able to see one another's cards

   The player to the left of the dealer begins.

C.1.5. Duration of the game

The game is played in a series of "rounds". A round is completed when each player has had one turn (turns can be quite lengthy).

To complete a game: with 2 players, play 20 rounds, with 3.. 16, and with 4.. 12.

Keep a note of the number of rounds played.

C.1.6. Winning the game

At the end of the last round, after the last Market Influence card has been played. the player with the highest total of assets in cash, and in shares at the final market prices, is the winner.

C.1.7. Objectives

Each player begins with a 450 Objective card. The figures given on the card represent the minimum value for each type of share that the player must hold to achieve his objective. Only three types of shares are given and the player must not hold any shares of the other two types when he claims his objective has been achieved.

This does not, of course, prevent the player from buying or selling any type of shares to make a profit at any stage of the game.

The value of the shares held is determined by multiplying the number of shares held by the ruling market price — e.g. 9 Mining at 12 each are worth 108.

A player can only achieve his objective after he has played the second Market Influence card of his turn, and after the Index prices have been adjusted accordingly — but before the next player begins his turn. This is the only time an objective can be achieved.

On achieving his objective the player declares his hand, which is then checked by the other players. If they are satisfied that his objective has been achieved they each pay him 25 and the Bank pays him 125. If there is insufficient money in the Bank the player must take the balance in any shares of his choice held by the Bank. The total value in cash and shares must not exceed 125. If necessary a small balance of cash may be left in the Bank.

If any player cannot pay his 25 he must hand over shares to the value of 50 to the one who has achieved his objective. Such shares may be selected by the player achieving his objective after inspecting the shares held by the defaulting player.

If shares are standing at 0 these may not be claimed. When all payments in cash or shares have been made, the one who achieved his objective places his Objective card at the bottom of the relevant pack. He then takes the top card from the next higher pack of Objective cards— e.g. if he achieves a 450 objective he places his old Objective card at the bottom of the 450 pack and then takes the top card from the 500 pack.

The game continues as before, with players working their way "upwards" from 450 objectives towards the 650 objectives —though they will not necessarily reach that point before the end of the game.

When a 650 objective is achieved, the player returns his first 650 card to the bottom of the pack and then takes another 650 from the top.

The achieving of an objective is simply one incident in a round, it does not end a round.

C.1.8. The turn

Each turn always consists of the following steps, strictly in the order given.

1. The player pays 50 to the Bank (this represents his trading costs).

   If a player is unable to pay 50 at the start of his turn he must pay 100 to the Bank during the course of his turn and before he plays his last Market influence card.

2. He plays any one of his Market Influence cards by placing it face up on the discard pile. He immediately replenishes his hand so that three Market Influence cards are held.

   The price Index markers must now be adjusted according to the instructions on the exposed card — e.g. if the card says "Stores up 2" the Stores column marker must be moved two lines up.

3. He may now buy and sell shares if he wishes, as explained under the headings "Buying" and "Selling" below.

4. Having concluded his transactions he ends his turn by playing one more Market Influence card. As usual he immediately replenishes his hand and the Price Index is again adjusted. The turn now passes to the next player.

C.1.9. Buying

If the player whose turn it is, wishes to buy a certain kind/s of share/s he proceeds as follows:

1. He bids for a number of shares at the ruling market price. The number and kind of shares must be stated — e.g. "Buy 9 Mining" or "Buy 2 Transport" etc.

2. The player to his left has the first option to accept the bid for the stated number of shares or for a smaller number. If this player does not satisfy the buyer's bid in full, the next player to the left has the next option to sell to the buyer the balance of the shares he wants — and so on. If a player does not wish to sell to the buyer he simply says "No" e.g.:

- Bid by player — "Buy 9 Mining"
- First Player — "Sell 5"
- Second Player — "No"
- Third Player — "Sell 4"
- and so the buyer gets the 9 he wants.

3. If the bid is not fully satisfied by any or all of the players, the buyer has the right to purchase the balance of the shares from the Bank — if they are available — at a price of 1 per share above the ruling market price. In this event he pays the money to the Bank.

4. It will sometimes happen that a player is unable to obtain alt the shares he wants.

C.1.10. Selling

If the player whose turn it is wishes to sell a certain kind/s of share/s he proceeds as follows:

1. He offers a number of shares at the ruling market price. The number and kind of shares must be stated e.g. "Sell 7 Property" or "Sell 25 Transport" etc.

2. The player on his left has the first option to accept the offer. If this player buys some but not all the shares offered, the next player has the option to buy the balance and so on.

3. If the offer is not taken up — or only partly taken up — by any or all of the players, the seller has the right to sell the balance of the shares to the Bank at a price of 1 per share below the ruling market price —and provided the Bank has sufficient cash. The Bank can only buy that number of shares for which it has sufficient cash.

4. It will sometimes happen that a player is unable to sell all the shares he wishes to. During his turn a player may deal in as many types of shares as he wishes, and as often as he wishes. The only exceptions to this rule are in the first and last rounds.

C.1.11. First and last rounds

In these two rounds only two types of shares may be dealt in during each player's turn.

C.1.12. End of the round

At the end of each round (i.e. when each player has had one turn) all the Market Influence cards from the pile and the discard must be shuffled together again. Cards held by the players are retained by them.

C.1.13. Share prices

Prices cannot go higher than 30 or lower than 0. The only exception is when the price stands at 30 and shares are bought from the Bank at 31 (as usual, 1 above the ruling market price). If shares are standing at 0 the Bank will give them away—without payment.

If a Market Influence card is played that would move the Index beyond the limits of 0 and 30 the card can be played but it will not have its full effect as the share prices stop at 0 and 30.

C.1.14. Dividend payments

There are five Dividend cards in the Market Influence pack. These cards can be played instead of Market Influence cards (for the purposes of the rules they are regarded as Market Influence cards). The dividends the players receive will depend on the current market price of the shares. When a Dividend card is played, all the players who hold shares of that particular type are eligible for payment. They all disclose their holdings and receive payment from the Bank as follows — number of shares held X the appropriate dividend as given on the Dividend card. For example — a Mining dividend is declared, the market price is 14 and a player holds 6 shares. The dividend card shows that the holder of shares whose market price is 13 — 20 receives 3 for each share, so he receives 6 x 3 = 18.

If the Bank has insufficient cash to pay the dividend to all the players holding shares — no dividend is paid at all.

C.1.15. Bankruptcy

A player who cannot meet his debts in full is out of the game. The cards he holds must be returned to the appropriate packs.

C.2. Manual traducido al castellano

C.2.1. El objetivo del juego

Este es un juego basado en las características de las operaciones con acciones y participaciones. Al igual que en la realidad, las estrategias de los inversores influyen decisivamente las tendencias del mercado.

Speculate no proporciona una situación en la que los jugadores ganan o pierden millones irreales, simula la posición del pequeño inversor tratando de aumentar el valor de sus participaciones por las cantidades modestas pero vitales que conoce en la vida real que puede esperar. Ahora, sin embargo, está en competencia con los demás, cada uno de los cuales tratarán de convertir la situación del mercado en su propio favor.

Habrá momentos en los que los jugadores tendrán la emoción de ver que con sus planes pueden ganar beneficios substanciales, pero si el mercado no sigue su camino estarán igualmente en una situación emocionante y desafiante en su lucha para mantener sus carteras y evitar la quiebra.

C.2.2. Estrategia

Los jugadores deben ser conscientes de que ellos-mismos pueden afectar radicalmente al mercado, produciendo condiciones a favor o adversas. Inevitablemente habrá períodos de depresión del mercado, así como ocasiones en las que sea boyante. Algunos jugadores son tentados, después de que tengan completas sus operaciones, a deprimir el mercado si pueden con el fin de reducir el valor de las participaciones de sus oponentes. Esto es a veces una estrategia perfectamente adecuada para adoptar pero siempre es prudente no hacer un mal uso!

Algunos jugadores pueden tener la tentación de esperar a que las acciones hayan caído a niveles muy bajos antes de hacer sus ofertas, con la esperanza de comprar barato. Deben ser conscientes de que en esta situación los otros jugadores no estarán dispuestos a vender y el mercado es probable que se convierta cada vez más estancado.

Durante el juego, los jugadores se les da "objetivos" - de determinadas acciones - para lograr. Cuando tienen éxito en alcanzar sus objetivos, los jugadores ganan un bono en efectivo. Sin embargo, los jugadores no deben concentrarse en el logro de sus objetivos hasta el punto de que no toman ventaja de las muchas oportunidades comerciales que se presentarán durante el curso del juego. Bien puede haber ocasiones en las que tendrá que pagar abandonando temporalmente el esfuerzo para lograr el "objetivo".

C.2.3. Equipo

1. 2.750 en billetes de banco (este total se establece deliberadamente con el fin de mantener el dinero)

2. Paquete de tarjetas de influencia de mercado (incluyendo las tarjetas 5 de dividendos)

3. Tarjetas Objetivo (10 de cada uno de 5 valores diferentes)

4. 50 Las acciones de cada uno de los 5 tipos (Certificados en unidades de 1, 2 y 3) (Minería, Tiendas, Propiedad, Transporte e Ingeniería)

5. Índice de Precios de Mercado (en la tapa de la caja)

6. 5 marcadores para la tabla de Precios

La caja del juego se utiliza en una forma que representa el mercado de valores general. Debe ser considerado como algo más que simplemente un Banco - pero "Banco" es una palabra fácil de usar, por lo que se utiliza en estas instrucciones.

C.2.4. Preparación

1. Nombrar a un banquero para controlar las relaciones con el Banco y otro jugador para controlar el índice de precios.

2. Vacíe la bandeja de todo su contenido. Separe las dos secciones de las calculadoras y doble sobre los extremos superiores de las tiras de selección. Estas bandas se deslizan a través de las columnas de cifras.

3. Coloque los cinco marcadores en el tablero índice del mercado para mostrar un precio de 10 para cada clase de acciones.

4. Dé a cada jugador 250 en efectivo y 8 acciones de cada tipo - es decir - 40 acciones en total.

5. Vuelva al Banco

   a) 300 en efectivo multiplicado por el número de jugadores.

   b) 2 acciones de cada tipo, multiplicado por el número de jugadores.

   Por ejemplo - si tres están jugando, el Banco tendrá 6 acciones de cada tipo y 900 en efectivo.

   Todas las demás acciones y billetes se deben poner a un lado - no se utilizan en el juego. Si 5 están jugando, por supuesto, se utilizarán todas las acciones y billetes.

6. Separar las tarjetas objetivo de los cinco valores diferentes. Baraje cada set y colóquelas boca abajo, en orden ascendente, en la bandeja. Otorgue una tarjeta de 450 Objetivo a cada jugador.

7. Baraje las cartas de influencia de mercado. Otorgue tres a cada jugador y coloque el resto hacia abajo en la bandeja.

   Los jugadores no deben ser capaces de ver una tarjeta de otra persona El jugador a la izquierda del repartidor comienza.

C.2.5. Duración del juego

El juego se desarrolla en una serie de "rondas". Se completa una ronda cuando cada jugador ha tenido un turno (los turnos pueden ser bastante largo). Para completar un juego: con 2 jugadores, jugar 20 rondas, con 3.. 16 y con 4.. 12.

Tome nota del número de rondas jugadas.

C.2.6. Ganar el juego

Al final de la última ronda, después de la tarjeta de Influencia de mercado se ha jugado. El jugador con el mayor total de activos en efectivo y en acciones al final del juego, es el ganador.

C.2.7. Objetivos

Cada jugador comienza con una tarjeta de Objetivo de 450. Las cifras que figuran en la tarjeta representan el valor mínimo para cada tipo de acción que el jugador tiene que tener para lograr su objetivo. Sólo se dan tres tipos de acciones y el jugador no debe contener las acciones de los otros dos tipos para lograr el objetivo.

Esto no significa, por supuesto, que el jugador no pueda comprar cualquier tipo de acciones para obtener un beneficio en cualquier etapa del juego.

El valor de las acciones en poder se determina multiplicando el número de acciones poseídas por el precio de mercado dominante - por ejemplo, 9 acciones de Minería a 12 cada uno valen 108.

Un jugador sólo puede alcanzar su objetivo después de que haya jugado la segunda tarjeta de influencia de mercado de su turno, y después de que los precios se han ajustado - pero antes de que el siguiente jugador comience su turno. Esta es la única vez que un objetivo puede ser alcanzado.

En la consecución de su objetivo, el jugador enseña su carta, que luego se comprobará por los otros jugadores. Si están satisfechos de que su objetivo se haya logrado, que cada uno le pagan 25 y el Banco le paga 125. Si no hay dinero suficiente en el banco el jugador debe tomar acciones cuyo valor sea el mismo del Banco. El valor total en efectivo y acciones no debe superar los 125. Si es necesario, un pequeño saldo de efectivo se puede dejar en el Banco.

Si algún jugador no puede pagar su 25 debe entregar acciones al valor de 50 para el que ha logrado su objetivo. Tales acciones pueden ser seleccionados por el jugador lograr su objetivo después de inspeccionar las acciones en poder del jugador de impago.

Si las acciones se están colocando a 0 estos no pueden ser reclamados. Una vez realizados todos los pagos en efectivo o en acciones, el que logró su objetivo coloca su tarjeta de Objetivo en la parte inferior del paquete correspondiente. A continuación, toma la carta superior del siguiente montón de cartas de objetivo - por ejemplo, si permite alcanzar un objetivo de 450 coloca su carta de Objetivo en la parte inferior del montón 450 y luego toma la carta superior del paquete de 500.

El juego continúa como antes, con los jugadores labran su camino "hacia arriba"de objetivos de 450 hacia los de 650 -aunque no alcanzarán necesariamente ese punto antes de la final de la partida.

Cuando se alcanza un objetivo 650, el jugador devuelve su carta de 650 a la parte inferior del montón y luego toma otra de 650 desde la parte superior.

El logro de un objetivo es simplemente un acto de en una ronda, que no termina dicha ronda.

C.2.8. El Turno

Cada turno siempre consta de los siguientes pasos, estrictamente en el orden indicado.

1. El jugador paga 50 al Banco (esto representa sus costes de negociación). Si un jugador no puede pagar el 50 al comienzo de su turno, debe pagar 100 al Banco durante el curso de su turno y antes de que juegue su última carta de influencia.

2. Usa cualquiera de sus tarjetas de influencia en el mercado, colocándola boca arriba en la pila de descartes. De inmediato se repone la mano para que tres cartas influencia de mercado se mantengan.

   Los marcadores de índice de precios ahora deben ajustarse de acuerdo con las instrucciones de la carta expuesta - por ejemplo, si la tarjeta dice "Tiendas suben 2" el marcador de columna de Tiendas debe mover dos líneas arriba.

3. Ahora puede comprar y vender acciones si lo desea, como se explica en los capítulos "Compraz "Venta".

4. Tras concluir sus operaciones, se debe jugar una carta influencia de mercado. Como de costumbre, inmediatamente vuelve a llenar su mano y el índice de precios se ajustará de nuevo. El turno ahora pasa al siguiente jugador.

C.2.9. Compra

Si el jugador que tiene el turno, desea comprar un determinado tipo de accion/es se procede de la siguiente manera:

1. Pide un número de acciones a precio de mercado dominante. El número y tipo de acciones deben indicarse - por ejemplo, "Comprar 9 Minería" o "Comprar 2 Transporte", etc.

2. El jugador a su izquierda tiene la primera opción de aceptar la compra por el número indicado de acciones o por un número más pequeño. Si este jugador no satisface la oferta del comprador en su totalidad, el siguiente jugador a la izquierda tiene la siguiente opción de vender al comprador el balance de las acciones que quiere - y así sucesivamente. Si un jugador no desea vender al comprador que simplemente dice "No" por ejemplo:

   - Haga una oferta por el jugador - "Comprar 9 Minería"
   - Primer jugador - "Vender 5"
   - Segundo Jugador - "No"
   - Tercer Jugador - "Vender 4"
   - Por lo que el comprador recibe el 9 que quiere.

3. Si la oferta no está completa con cualquiera o todos los jugadores, el comprador tiene el derecho de comprar el resto de las acciones del Banco - si están disponibles - a un precio de 1 por acción por encima del precio de mercado dominante. En este caso se paga el dinero al Banco.

4. A veces sucede que un jugador no puede obtener todas las acciones que quiere.

C.2.10. Venta

Si el jugador que tiene el turno desea vender un tipo determinado/s de accion/es se procede de la siguiente manera:

1. Él ofrece una serie de acciones al precio de mercado. El número y tipo de acciones deben indicarse por ejemplo "Vender 7 Propiedad" o "vender 25 Transporte", etc.

2. El jugador a su izquierda tiene la primera opción de aceptar la oferta. Si este jugador compra algunas pero no todas las acciones ofrecidas, el siguiente jugador tiene la opción de comprar el equilibrio y así sucesivamente.

3. Si la oferta no es aceptada - o sólo en parte ocupado - por cualquiera o todos los jugadores, el vendedor tiene el derecho de vender el resto de las acciones al Banco a un precio de 1 por acción por debajo del precio de mercado dominante - y siempre que el Banco cuente con suficiente dinero en efectivo. El Banco sólo puede comprar ese número de acciones para las que tenga suficiente dinero en efectivo.

4. A veces sucede que un jugador no puede vender todas las acciones que él desea. Durante su turno un jugador puede hacer frente en tantos tipos de acciones a su antojo, y tantas veces como lo desee. Las únicas excepciones a esta regla están en las primeras y últimas rondas.

C.2.11. Rondas primera y la última

En estas dos rondas sólo dos tipos de acciones podrán ser negociados durante el turno de cada jugador.

C.2.12. Fin de la ronda

Al final de cada ronda (es decir, cuando todos los jugadores hayan jugado su turno) todas las tarjetas de influencia de mercado del montón y el descarte debe ser barajado juntos de nuevo. Las cartas poder de los jugadores son mantenidas.

C.2.13. Precios de las acciones

Los precios no pueden subir más de 30 o inferior a 0. La única excepción es cuando el precio se sitúa en el 30 y se compran acciones del Banco a 31 (como de costumbre, 1 por encima del precio de mercado dominante). Si las acciones están de pie en 0 el Banco les regalará-sin pago.

Si se juega una carta de Influencia de mercado que movería el índice más allá de los límites de 0 y 30 la tarjeta se puede jugar pero no va a tener su efecto completo como los precios de las acciones se detienen en 0 y 30.

C.2.14. El pago de dividendos

Hay cinco cartas de dividendos en el montón de cartas de influencia de mercado. Estas cartas se pueden jugar en lugar de tarjetas de influencia en el mercado (se consideran como las cartas de influencia de mercado). Los dividendos que los jugadores reciben dependerán del precio actual de las acciones del mercado. Cuando se juega una carta de dividendos, todos los jugadores que posean acciones de ese tipo particular son elegibles para el pago. Todos ellos dan a conocer sus tenencias y reciben el pago por parte del Banco de la siguiente manera - Número de acciones X dividendo apropiado especificado en la tarjeta de dividendos.

Por ejemplo - si se juega una carta de dividendo de Minería y el precio de mercado es de 14 y un jugador tiene 6 acciones. La tarjeta de dividendo acredita que el titular de las acciones cuyo mercado precio es 13 a 20 recibe 3 por cada acción, por lo que recibe 6 x 3 = 18.

Si el Banco no tiene efectivo suficiente para pagar el dividendo a todos los jugadores que posean acciones, no se paga ningún dividendo.

C.2.15. Bancarrota

Un jugador que no puede cumplir con sus deudas en su totalidad está fuera del juego. Las cartas que posee se deben devolver a los montones apropiados.

D

Código abierto

D.1. Licencia

El proyecto se ha llevado a cabo siguiendo la licencia [14, MIT]. Es una licencia de software libre permisiva y ha sido empleada por el *Instituto Tecnológico de Massachusetts* a lo largo de su historia.

> Copyright (c) 2014 Javier Segura
>
> Se concede permiso por la presente, de forma gratuita, a cualquier persona que obtenga una copia de este software y de los archivos de documentación asociados (el "Software"), para utilizar el Software sin restricción, incluyendo sin limitación los derechos de usar, copiar, modificar, fusionar, publicar, distribuir, sublicenciar, y/o vender copias de este Software, y para permitir a las personas a las que se les proporcione el Software a hacer lo mismo, sujeto a las siguientes condiciones:
>
> El aviso de copyright anterior y este aviso de permiso se incluirán en todas las copias o partes sustanciales del Software.
>
> EL SOFTWARE SE PROPORCIONA "TAL CUAL", SIN GARANTÍA DE NINGÚN TIPO, EXPRESA O IMPLÍCITA, INCLUYENDO PERO NO LIMITADO A GARANTÍAS DE COMERCIALIZACIÓN, IDONEIDAD PARA UN PROPÓSITO PARTICULAR Y NO INFRACCIÓN. EN NINGÚN CASO LOS AUTORES O TITULARES DEL COPYRIGHT SERÁN RESPONSABLES DE NINGUNA RECLAMACIÓN, DAÑOS U OTRAS RESPONSABILIDADES, YA SEA EN UN LITIGIO, AGRAVIO O DE OTRO MODO, QUE SURJA DE O EN CONEXIÓN CON EL SOFTWARE O EL USO U OTRO TIPO DE ACCIONES EN EL SOFTWARE.

D.2. Código fuente

El código fuente puede ser consultado por cualquiera gracias a la licencia antes mencionada. Está disponible en la plataforma [15, Github] y podrá ser

descargado y ejecutado sin tener que avisar al autor una vez presentado ante el tribunal de la Universidad.

D.2.1. Cómo obtener y ejecutar el código

Para obtener el código hay que usar Git. Para poder ejecutar el software tenemos que tener los siguientes paquetes instalados en el sistema:

1. *Mongo* [17] ejecutándose en localhost
2. *Node.js* [18] instalado en el sistema
3. *Grunt* [7] instalado en el sistema

Se puede obtener y ejecutar siguiendo el siguiente proceso en cualquier terminal Unix.

```
$ git clone git@github.com:jsegura/speculate.git
$ cd speculate
$ npm install
$ scripts/repopulate-db
$ grunt serve
```

Siguiendo los pasos tendremos disponible en *http://localhost:3000* un servidor de speculate corriendo y listo para funcionar con usuarios ya cargados.